IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR UNITED STATES LETTERS PATENT

MECHANISM TO CONTROL THE ALLOCATION OF AN N-SOURCE SHARED BUFFER

By:

Michael S. Bertone
119 Roundtop Road
Marlborough, MA 01752
Citizenship: U.S.A.

Richard E. Kessler
30 Thestland Dr.
Shrewsbury, MA 01545
Citizenship: U.S.A.

David H. Asher
415 Central Turnpike
Sutton, MA 01590
Citizenship: U.S.A.

Steve Lang
359 Taylor Road
Stow, MA 01775
Citizenship: U.S.A.

Express Mail Label No. EL705960437US

MECHANISM TO CONTROL THE ALLOCATION OF AN N-SOURCE SHARED BUFFER

CROSS-REFERENCE TO RELATED APPLICATIONS

This application relates to the following commonly assigned co-pending applications entitled:

"Scan Wheel – An Apparatus For Interfacing A High Speed Scan-Path With A Slow Speed Tester," Serial No. __________, filed August 31, 2000, Attorney Docket No. 1662-23700; "Rotary Rule And Coherence Dependence Priority Rule," Serial No. __________, filed August 31, 2000, Attorney Docket No. 1662-27300; "Speculative Scalable Directory Based Cache Coherence Protocol," Serial No. __________, filed August 31, 2000, Attorney Docket No. 1662-27400;

"Scalable Efficient IO Port Protocol," Serial No. __________, filed August 31, 2000, Attorney Docket No. 1662-27500; "Efficient Translation Buffer Miss Processing For Applications Using Large Pages In Systems With A Large Range Of Page Sizes By Eliminating Page Table Level," Serial No. __________, filed August 31, 2000, Attorney Docket No. 1662-27600; "Fault Containment And Error Recovery Techniques In A Scalable Multiprocessor," Serial No. __________, filed August 31, 2000, Attorney Docket No. 1662-27700; "Speculative Directory Writes In A Directory Based CC-Non Uniform Memory Access Protocol," Serial No. __________, filed August 31, 2000, Attorney Docket No. 1662-27800; "Special Encoding Of Known Bad Data," Serial No. __________, filed August 31, 2000, Attorney Docket No. 1662-27900; "Broadcast Invalidate Scheme," Serial No. __________, filed August 31, 2000, Attorney Docket No. 1662-28000; "Mechanism To Keep All Pages Open In A DRAM Memory System," Serial

No. __________, filed August 31, 2000, Attorney Docket No. 1662-28100; "Programmable DRAM Address Mapping Mechanism," Serial No. __________, filed August 31, 2000, Attorney Docket No. 1662-28200; "Mechanism To Enforce Memory Read/Write Fairness, Avoid Tristate Bus Conflicts, And Maximize Memory Bandwidth," Serial No. __________, filed August 31, 2000, Attorney Docket No. 1662-29200; "An Efficient Address Interleaving With Simultaneous Multiple Locality Options," Serial No. __________, filed August 31, 2000, Attorney Docket No. 1662-29300; Serial No. __________, filed August 31, 2000, Attorney Docket No. 1662-29400; "Method And System For Absorbing Defects In High Performance Microprocessor With A Large N-Way Set Associative Cache," Serial No. __________, filed August 31, 2000, Attorney Docket No. 1662-29500; "A Method For Reducing Directory Writes And Latency In A High Performance, Directory-Based, Coherency Protocol," Serial No. __________, filed August 31, 2000, Attorney Docket No. 1662-29600; "Mechanism To Reorder Memory Read And Write Transactions For Reduced Latency And Increased Bandwidth," Serial No. __________, filed August 31, 2000, Attorney Docket No. 1662-30800; "Look-Ahead Mechanism To Minimize And Manage Bank Conflicts In A Computer Memory System," Serial No. __________, filed August 31, 2000, Attorney Docket No. 1662-30900; "Resource Allocation Scheme That Ensures Forward Progress, Maximizes Utilization Of Available Buffers And Guarantees Minimum Request Rate," Serial No. __________, filed August 31, 2000, Attorney Docket No. 1662-31000; "Input Data Recovery Scheme," Serial No. __________, filed August 31, 2000, Attorney Docket No. 1662-31100; "Fast Lane Prefetching," Serial No. __________, filed August 31, 2000, Attorney Docket No. 1662-31200; "Mechanism For Synchronizing Multiple Skewed Source-Synchronous Data Channels With Automatic Initialization Feature," Serial No. __________, filed August 31, 2000, Attorney Docket No. 1662-31300; and "Chaining Directory Reads And Writes To Reduce DRAM

Bandwidth In A Directory Based CC-NUMA Protocol," Serial No. __________, filed August 31, 2000, Attorney Docket No. 1662-31500, all of which are incorporated by reference herein.

STATEMENT REGARDING FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT

Not applicable.

BACKGROUND OF THE INVENTION

Field of the Invention

The present invention generally relates to the distribution of buffer space between multiple sources. More particularly, the invention relates to a fair and efficient method of controlling allocation of a shared buffer pool.

Background of the Invention

In computer systems and networks, buffers are a convenient means of storing commands, requests, and data that are in transit from one location to another. Buffers may be used in a variety of applications. They may be used in network switching devices to temporarily hold data packets while network congestion subsides or while the switch determines the location to which the data must be forwarded. It is not uncommon for network switches to manage traffic for a plurality of sources. Buffers may also be used in memory and data allocation. An example of the latter would be a data read/write request buffer that must allocate requests from multiple sources. A common problem in systems using a shared buffer space is signal traffic that creates congestion and may lead to buffer overflow and monopolization by one or more of the buffer sources.

Ideally, a system comprising a buffer with multiple sources should accomplish several tasks. First, the system should not deliver data, commands, or requests to the buffer if the buffer does not have any free space. This prevents data loss or packet drops which may require that the data packet be re-sent resulting in even greater bandwidth loss than simply holding the data until buffer space becomes available. Secondly, access to buffer space by the multiple sources should preferably be allocated in a fair manner. The sources should have fair access to the buffer so a source does not become backlogged while other sources are able to deliver data freely. This does not necessarily imply that the allocation needs to be equal for each source. For instance, one source may be given priority over the others. However, even in this scenario, it is important to prevent complete monopolization by the source that has priority.

One conventional solution to the problem of fair allocation of a shared buffer space is hard partitioning of the buffer space. For example, if a buffer has 16 data spaces and the buffer is shared among 4 sources, each source may be allocated four buffer slots. This method of allocation is certainly fair but may be horribly inefficient. If one of the sources has a string of data that ideally could be burst to the buffer, congestion may occur because the source only has four buffer spaces available. The remaining 12 buffers could be used to hold the burst of data, but instead may lie dormant because of the hard partitioning.

If prior knowledge exists about the type of traffic that can be expected from the sources, the hard partitioning may be altered to allocate more buffer space to one source or another. For instance, in the example given above, seven buffer spaces may be allocated to one source while the other three sources are allocated three spaces each. This allocation may alleviate congestion for the prioritized source, but does not prevent the burst congestion for of the other sources. In either case, hard partitioning tends to preclude use of at least a fixed percentage of the buffer space unless all sources are continuously accessing the buffer.

Another conventional solution to the problem of fairly and efficiently allocating buffer space is with stop and go commands issued by the buffer. In this type of system, the buffer is configured to keep track of available spaces within the buffer. During normal operation with light traffic, each source receives a "go" signal from the buffer indicating that buffer space is available. As buffer space becomes limited, the buffer may send "stop" signals to the individual sources to halt data transmission to the buffer. This approach offers better use of the buffer space because the sources are not limited to a fixed percentage of the buffer space. However, some risk is involved in this type of embodiment because a finite time exists between the moment the buffer sends out a stop command and the moment the source receives and responds to the stop command. During this finite time, it is possible for additional data to be transmitted to the buffer from the various sources. If the buffer was sufficiently close to being full and enough data was sent to the buffer before the stop commands were received by the sources, buffer overflow may occur and data may be lost. To prevent this, stop commands are usually sent well in advance of the buffer filling to capacity. Thus, if all buffer sources are bursting data to the buffer, the stop command is preferably timed so that the sources stop sending data to the buffer before the buffer capacity is filled. Unfortunately, the side effect of sending the stop commands early is that the maximum capacity of the buffer will not be used when the buffer sources are not simultaneously sending bursts of data. The stop/go command solution to this buffer allocation problem is an improvement over the hard partitioning solution, but presents problems of either overflow or inefficient use of the whole buffer.

It is desirable therefore to develop a fair and efficient means of allocating buffer space among several sources. The allocation scheme preferably prevents monopolization by any of the buffer sources. The allocation method also preferably takes advantage of the full capacity of the buffer space without the possibility of buffer overflow. The system may advantageously be applied to a plurality of buffer sources and may also be applied to a variety of applications.

BRIEF SUMMARY OF THE INVENTION

The problems noted above are solved in large part by a method and apparatus for ensuring fair and efficient use of a shared memory buffer. The invention uses a credit-based allocation scheme to prevent monopolization by one or more sources and permits efficient use of the entire buffer. A preferred embodiment comprises a shared memory request buffer in a multi-processor computer system. The shared memory request buffer is used to store requests from different processors. Memory requests from a local processor are delivered to the local memory controller by a cache control unit. Requests from other processors are delivered to the memory controller by an interprocessor router. The memory controller allocates the memory requests in a shared buffer using a credit-based allocation scheme. The cache control unit and the interprocessor router are each assigned a number of credits. Each must pay a credit to the memory controller when a request is allocated to the shared buffer. If a source does not have any available credits, that source may not send a request to the shared buffer. The number of credits assigned to each source is sufficient to enable each source to deliver an uninterrupted burst of memory requests to the buffer without having to wait for credits to return from the buffer. The total number of credits assigned to the sources is preferably small compared to the overall size of the buffer. If the number of filled spaces in the shared buffer is below a threshold, the buffer immediately returns the credits to the source from which the credit and memory request arrived. If the number of filled spaces in the shared buffer is above a threshold, the buffer holds the credits and returns a single credit in a round-robin manner only when a space in the shared buffer becomes free. The buffer threshold is the point when the number of free spaces available in the buffer is equal to the total number of credits assigned to the cache control unit and the interprocessor router. Since credits are not freely returned as the buffer gets full and since there are never any more credits available than spaces in the buffer, the buffer may reach capacity, but will not overflow.

BRIEF DESCRIPTION OF THE DRAWINGS

For a detailed description of the preferred embodiments of the invention, reference will now be made to the accompanying drawings in which:

Figure 1 shows a system diagram of a plurality of microprocessors coupled together;

Figures 2 shows a block diagram of a microprocessor memory controller in which the preferred embodiment is implemented;

Figure 3 shows a schematic representation of the credit-based allocation of the memory request buffer in the memory controller of Figure 2 operating in a light load condition; and

Figure 4 shows the memory request buffer of Figure 3 operating in a heavy load condition.

NOTATION AND NOMENCLATURE

Certain terms are used throughout the following description and claims to refer to particular system components. As one skilled in the art will appreciate, computer companies may refer to a component by different names. This document does not intend to distinguish between components that differ in name but not function. In the following discussion and in the claims, the terms "including" and "comprising" are used in an open-ended fashion, and thus should be interpreted to mean "including, but not limited to...". Also, the term "couple" or "couples" is intended to mean either an indirect or direct electrical connection. Thus, if a first device couples to a second device, that connection may be through a direct electrical connection, or through an indirect electrical connection via other devices and connections.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The preferred embodiment of the shared buffer allocation scheme is directed toward application in a memory request buffer used in a multi-processor computer system. While the preferred embodiment may be implemented in a variety of applications, a detailed description in the context of the multi-processor computer system is provided herein. The descriptions herein are not intended to limit the scope of the claim limitations set forth in this specification.

Referring now to Figure 1, in accordance with the preferred embodiment of the invention, computer system 90 comprises one or more processors 100 coupled to a memory 102 and an input/output ("I/O") controller 104. As shown, computer system 90 includes twelve processors 100, each processor coupled to a memory and an I/O controller. Each processor preferably includes four ports for connection to adjacent processors. The interprocessor ports are designated "North," "South," "East," and "West" in accordance with the well-known Manhattan grid architecture also known as a crossbar interconnection network architecture. As such, each processor 100 can be connected to four other processors. The processors on both ends of the system layout wrap around and connect to processors on the opposite side to implement a 2D torus-type connection. Although twelve processors 100 are shown in the exemplary embodiment of Figure 1, any desired number of processors (*e.g.*, 256) can be included. For purposes of the following discussion, the processor in the upper, left-hand corner of Figure 1 will be discussed with the understanding that the other processors 100 are similarly configured in the preferred embodiment.

As noted, each processor preferably has an associated I/O controller 104. The I/O controller 104 provides an interface to various input/output devices such as disk drives 105 and 106, as shown in the lower, left-hand corner of Figure 1. Data from the I/O devices thus enters the 2D torus via the I/O controllers.

Each processor also, preferably, has an associated memory 102. In accordance with the preferred embodiment, the memory 102 preferably comprises RAMbus^{TM} memory devices, but other types of memory devices can be used, if desired. The capacity of the memory devices 102 can be any suitable size. Further, memory devices 102 preferably are implemented as Rambus Interface Memory Modules ("RIMM"). To aid in the control of distributed memory in the multiprocessor system, each processor includes a memory manager and directory structure for the local memory source. A preferred embodiment of the memory controller is shown in Figure 2.

In general, computer system 90 can be configured so that any processor 100 can access its own memory 102 and I/O devices, as well as the memory and I/O devices of all other processors in the system. Preferably, the computer system may have physical connections between each processor resulting in low interprocessor communication times and improved memory and I/O device access reliability. If physical connections are not present between each pair of processors, a pass-through or bypass path is preferably implemented in each processor that permits accesses to a processor's memory and I/O devices by another processor through one or more pass-through processors.

Referring now to Figure 2, each processor 100 preferably includes an L2 instruction and data cache control unit ("Cbox") 170, an L2 data cache 180, a memory controller ("Zbox") 190, and an interprocessor and I/O router unit ("Rbox") 200. The following discussion describes each of these units.

Each of the functional units 170, 180, 190, 200 contains control logic that communicates with the control logic of other functional units as shown. Other functional units may exist within the processor 100, but have been omitted from Figure 1 for clarity in the description of the preferred embodiment. For example, the processor may further comprise integer and floating-point execution units, a memory reference unit, and an instruction fetch, issue and retire unit.

The L2 instruction and data cache control unit ("Cbox") 170 controls the L2 instruction and data cache 180 and handles data accesses from other functional units in the processor, other processors in the computer system, or any other devices that might need data out of the L2 cache.

The interprocessor and I/O router unit ("Rbox") 200 provides the interfaces to as many as four other processors and one I/O controller 104 (Figure 1). The inter-processor interfaces are designated as North ("N"), South ("S"), East ("E"), and West ("W") and provide two-way communication between adjacent processors.

Processor 100 preferably includes at least one RAMbus™ memory controller 190 ("Zbox"). The Zbox 190 controls 4 or 5 channels of information flow with the main memory 102 (Figure 1). The Zbox 190 preferably includes a back end 193, a middle mapper 192, and a front-end directory in flight table ("DIFT") 191. The middle mapper 192 maps the physical address into RAMbus™ device format by device, bank, row, and column. The middle mapper 192 also maintains an open-page table to track all open pages and to close pages on demand if bank conflicts arise. The mapper 192 also schedules RAMbus™ transactions such as timer-base request queues. The Zbox back end 193 preferably packetizes the address, control, and data into RAMbus™ format and provides the electrical interface to the RAMbus™ devices themselves.

The front-end DIFT 191 performs a number of functions such as managing the processor's directory-based memory coherency protocol, processing request commands from the Cbox 170 and

Rbox 200, sending forward commands to the Rbox 200, sending response commands to and receiving packets from the Cbox 170 and Rbox 200.

The DIFT 191 also comprises a shared request buffer for tracking up to thirty-two in-flight transactions. These transaction requests or packets are received from the Cbox 170 or the Rbox 200. When a request comes from either source, the DIFT must allocate an entry in the 32-space buffer. The requests from the Cbox 170 are from the local processor chip whereas requests from the Rbox 200 are off chip requests. Since each processor in the multi-processor system shown in Figure 1 is identical, a Cbox 170 request should carry the same priority as a request from the Rbox 200. It is important therefore to allocate space in the shared DIFT buffer in a fair and efficient manner.

Figure 3 represents a simplified schematic of the preferred embodiment of a credit allocation scheme for the DIFT buffer. The DIFT 191 preferably includes a shared request buffer 300 and credit counters 310, 320, 330. The Cbox 170 preferably includes a credit counter 340. Likewise, the Rbox includes a credit counter 350. The credit counters in the DIFT 191, Cbox 170 and Rbox 200 determine when the Cbox 170 and Rbox 200 may deliver a request to the DIFT 191. For both the Cbox 170 and Rbox 200, a request may only be delivered to the DIFT 191 when a credit is available in the local credit counter 340 and 350, respectively. A credit from Cbox counter 340 is spent (i.e., given up to the DIFT credit counter 320) by the Cbox 170 when a request is sent to the DIFT buffer 300. Similarly, a credit from Rbox counter 350 is spent (*i.e.*, given up to the DIFT credit counter 330) by the Rbox 200 when a request is sent to the DIFT buffer 300.

In the preferred embodiment, credits are returned from the DIFT 191 to the Cbox 170 and Rbox 200 in two distinct manners. The method of returning credits depends on the region of operation for the DIFT buffer 300. The first occurs under light loads when the DIFT buffer 300 is relatively empty. This condition is met when the number of occupied buffer spaces is below a threshold 360. In the example shown in Figure 3, the DIFT buffer 300 only has a few buffer spaces filled. In this situation, the DIFT 191 immediately returns credits to the source from which a request arrives. The Cbox 170 and Rbox 200 will therefore have all or nearly all of their credits in light load conditions.

The number of credits assigned to the Cbox 170 and Rbox 200 are based, in part, on credit round trip times. Since credits are returned immediately during light load situations, the total time required to transmit a credit from the Cbox 170 to the DIFT 191 and for the DIFT 191 to return the credit to the Cbox 170 may be determined. The number of credits given to the Cbox 170 is based not only on this round trip time, but also on the speed with which the Cbox 170 may deliver a burst of requests. The preferred embodiment gives enough credits so that if the Cbox 170 has enough memory requests from the processor, it may continuously deliver these requests without having to wait for credits to return from the DIFT 191. Consider for example that the Cbox 170 has a string of requests that need to be sent yet only has 4 credits. Ideally if the Cbox bursts these requests one after another as quickly as it can, then by the time the Cbox is ready to transmit the fifth request, the credit from the first request has preferably arrived back at the Cbox 170. This credit may then be used to transmit the fifth request. Subsequent requests may then be transmitted with other credits as they arrive back at the Cbox 170. This process may continue as long as the number of occupied spaces in the DIFT buffer remains under the threshold 360.

The number of credits given to the Rbox 200 is determined in the same way as for the Cbox 170. The round trip times between the two sources should not be assumed to be the same. A source with longer credit round trip times will necessarily require more credits to guarantee uninterrupted burst requests. Thus the number of credits assigned to the Rbox 200 will probably, though not necessarily, differ from the number of credits given to the Cbox 170. In the system shown in Figure 3, the Cbox counter 340 holds four credits while the Rbox counter 350 holds five credits. The actual number of credits given to each source in the preferred embodiment is system specific and will depend on round trip times between source and buffer.

The second manner in which the DIFT 191 returns credits to the Cbox 170 and Rbox 200 occurs under heavier loads when the DIFT buffer 300 is relatively full. The threshold 360 between these two conditions is preferably defined as the difference between the size of the DIFT buffer 300 and the number of credits distributed among the sources. In the system shown in Figure 3, the threshold occurs when there are only nine buffer spaces left. When the DIFT buffer 300 is filled above this threshold, the DIFT 191 will hold credits until a buffer space is freed. This situation is depicted in Figure 4.

In Figure 4, the DIFT buffer 300 is filled above the threshold 360 and therefore, the DIFT 191 holds onto the credits spent by the Cbox 170 and Rbox 200. The credits held from the Cbox 170 and Rbox 200 are stored in counters 320 and 330, respectively. In this region of operation, the DIFT buffer counter 310 indicates the number of DIFT buffer spaces available. The threshold 360 is set so that the number of outstanding credits never exceeds the number of free spaces in the DIFT buffer 300. Thus, if the Cbox 170 and Rbox 200 send requests until all credits are gone, the DIFT buffer will be full. The preferred embodiment may therefore advantageously use the entire DIFT buffer 300 without running the risk of buffer overflow.

In this second region of operation, each time a space in the DIFT buffer 300 becomes available, credits are preferably returned to the Cbox 170 or Rbox 200. To ensure fairness, the credits are returned in a random, equally probable round-robin fashion to those sources which have spent credits. A suitable method of returning the credits may be to simply alternate credit returns.

Other embodiments exist where statistical methods are used to determine which source receives a credit. For instance, if priority needs to be given to the Cbox 170, the credit returns may be based on a random generator that is statistically skewed to return more credits to the Cbox 170 than the Rbox 200. The preferred embodiment however, returns credits on an equally likely basis to ensure fairness between the sources. It should also be noted that when the system operates in the first region of operation (during light loads), fairness between the sources is guaranteed because credits are returned immediately.

An alternative embodiment exists whereby the credit allocation scheme may be used to send request or response commands to the DIFT buffer 300. In this embodiment, the Cbox 170 and Rbox 200 are assigned credits as discussed above, but each must reserve a final credit for a response only. Thus, all credits in the Cbox 170 or Rbox 200 may be spent in issuing requests or responses to the DIFT buffer 300, but the last credit must be spent on a response.

The above discussion is meant to be illustrative of the principles and various embodiments of the present invention. Numerous variations and modifications will become apparent to those skilled in the art once the above disclosure is fully appreciated. For example, the shared buffer space may be extended to include three or more buffer sources. It is intended that the following claims be interpreted to embrace all such variations and modifications.